Bolt Beranek and Newman Inc.

Report No. 4868

Combined Quarterly Technical Report No. 24

SATNET Development and Operation
Pluribus Satellite IMP Development
Remote Site Maintenance
Internet Operations and Maintenance
Mobile Access Terminal Network
TCP for the HP3000
TCP for VAX-UNIX

February 1982

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMBINED QUARTERLY TECHNICAL REPORT No. 24		5. TYPE OF REPORT & PERIOD COVERED 11/1/81 to 1/31/82
		6. PERFORMING ORG. REPORT NUMBER 4868
7. AUTHOR(s) J. F. Haverty		8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0353 & 0214 N00039-78-C-0405 N00039-80-C-0664 N00039-81-C-0408
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order Nos. 3214 and 3175.17
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE February 1982
		13. NUMBER OF PAGES 76
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DSSW Room 1D, The Pentagon Washington, DC 20310	NAVELEX Washington, DC 20360	15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE/DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer networks, packets, packet broadcast, satellite communication, gateways, Transmission Control Protocol, UNIX, Pluribus Satellite IMP, Remote Site Module, Remote Site Maintenance, shipboard communications, VAX, ARPANET, Internet.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Quarterly Technical Report describes work on the development of and experimentation with packet broadcast by satellite; on development of Pluribus Satellite IMPs; on a study of the technology of Remote Site Maintenance; on Internetwork monitoring; on shipboard satellite communications; and on the development of Transmission Control Protocols for the HP3000 and VAX-UNIX.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4868

COMBINED QUARTERLY TECHNICAL REPORT NO. 24

SATNET DEVELOPMENT AND OPERATION
PLURIBUS SATELLITE IMP DEVELOPMENT
REMOTE SITE MAINTENANCE
INTERNET OPERATIONS AND MAINTENANCE
MOBILE ACCESS TERMINAL NETWORK
TCP FOR THE HP3000
TCP FOR VAX-UNIX

February 1982

This research was supported by the Defense Advanced Research Projects Agency under the following contracts:

✓N00039-78-C-0405, ARPA Order No. 3175.17
✓MDA903-80-C-0353, ARPA Order No. 3214
MDA903-80-C-0214, ARPA Order No. 3214
✓N00039-80-C-0664
✓N00039-81-C-0408

Submitted to:

Director
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Attention: Program Management

Table of Contents

1 INTRODUCTION

This Quarterly Technical Report is the current edition in a series of reports which describe the work being performed at BBN in fulfillment of several ARPA work statements. This QTR covers work on several ARPA-sponsored projects including (1) development and operation of the SATNET satellite network; (2) development of the Pluribus Satellite IMP; (3) Remote Site Maintenance activities; (4) Internet Operations, Maintenance, and Development; (5) development of the Mobile Access Terminal Network; (6) TCP for the HP3000; and (7) TCP for the VAX-UNIX. This work is described in this single Quarterly Technical Report with the permission of the Defense Advanced Research Projects Agency. Some of this work is a continuation of efforts previously reported on under contracts DAHC15-69-C-0179, F08606-73-C-0027, F08606-75-C-0032, MDA903-76-C-0214, MDA903-76-C-0252, and N00039-79-C-0386.

2 SATNET DEVELOPMENT AND OPERATION

As part of our participation in the Atlantic Packet Satellite Experiment (SATNET) during the last quarter, we installed the first BBN C/30 communications processor for field site operations as a Satellite IMP. This installation and the overall SATNET hardware maintenance operations are described in the following sections. Some of our other activities are described below.

After the release of new software into the BBN gateway, we discovered that the Etam Satellite IMP could not be loaded through the VDH circuit from the BBN gateway. Emergency software reloads required the IMP-to-IMP circuit between the Etam Satellite IMP and the SDAC IMP to be configured VDH temporarily. The problem was traced to a deficiency in the protocol between gateway and loader, although we also found a bug in the SATNET VDH Loader/Dumper packet-core module causing numerous packet-core retransmissions.

The deficiency resulted in the gateway declaring all SATNET operations down and discarding all loading packets destined for the Satellite IMP when no response to gateway interface probe messages was elicited from the SATNET VDH Loader/Dumper. To overcome this deficiency, the gateway-loader protocol has been modified to explicitly identify loader operations; namely, the

SATNET VDH Loader/Dumper sets a flag in the Restart/Request and Restart/Complete messages. Upon receiving a Restart/Request or a Restart/Complete with the flag set, the gateway will inhibit sending interface probes and will automatically declare the circuit to be up. Upon receiving a Restart/Request or a Restart/Complete with the flag reset, the gateway will proceed to invoke the normal interface up/down mechanism.

In order to rearrange the computational loading on the USC-ISI ARPANET host computers, we were required to move the directory SATIMP from the USC-ISIE TOPS-20 to the USC-ISID TOPS-20. This directory contains the SATNET monitoring programs, critical to the operation of the SATNET Network Operations Center (NOC). SATNET experimenters use this directory also.

Because of the decommission of the ARPANET host computer BBN-TENEX-E, we were required to move the SATNET project directories from BBN-TENEX-E to BBN-TENEX-C. These directories contain the source software for the Satellite IMP program and other SATNET ancillary programs, such as SATNET Loader/Dumpers. SATNET related software is currently cross assembled in these directories. The long-term goal, however, is to transfer the project directories to a UNIX (registered trademark of Bell Laboratories) computer. Toward that goal we have begun the conversion of Satellite IMP source software to a form compatible with a C/30 cross assembler currently residing in BBN-UNIX machines. Backroom debugging of this assembled software has occupied a significant amount of our time.

The dedicated satellite circuit between the SDAC IMP and the NORSAR TIP (ARPANET Line #41) has been permanently removed from service, isolating the NORSAR TIP from ARPANET. Only internetwork communications through SATNET will be available to former users of the NORSAR TIP requiring access to ARPANET.

2.1 C/30 Satellite IMP Installation

During the last quarter, a BBN C/30 communications processor was installed at Communications Satellite Corporation (COMSAT) Laboratories in Clarksburg, Maryland, replacing the Honeywell 316 general purpose minicomputer serving as a Satellite IMP. Whereas from inception Satellite IMPs in MATNET were designed with C/30s, this installation represents the first time ever that a C/30 has served as a Satellite IMP in SATNET. Connectivity between Clarksburg and the other Satellite IMPs on SATNET through the INTELSAT IV-A satellite remains untested, though, since the COMSAT Unattended Earth Terminal is non-functional with SATNET.

We have established the convention that Satellite IMP version numbers shall begin with a 4 instead of a 3 to indicate

C/30 hardware, while software compatibility is indicated by the second and third parts of the version number. Thus, version numbers of Satellite IMP software currently in the field are 3.4:1 for sites Etam, Goonhilly, and Tanum, and 4.4:1 for site Clarskburg.

The following summarizes the problems dealt with during the C/30 installation process. A flawed I/O bus terminator precluded diagnostic operations, until we discovered reversing this terminator enabled normal C/30 operations. Although data signal inversion is required, the original configuration for the RS-232-C daughterboard mounted above the C/30 universal I/O board caused the C/30 to transmit with one polarity and receive with another. Reversing the jumpers on one side of the board corrected the problem, allowing VDH connectivity to be established on the circuit to ARPANET IMP #71. Afterwards we were able to employ all the SATNET monitoring programs on TENEX/TOPS-20 ARPANET hosts for fault diagnosis.

Because the software written on the Digital Equipment Corporation (DEC) TU-58 tape cassette for loading the C/30 Satellite IMP was missing a previously identified patch to provide internal clocking for the Command Monitoring Module (CMM) interface of the Packet Satellite Project (PSP) terminal, the patch had to be manually inserted after each reload. In the absence of a configuration PROM identifying the Clarksburg Satellite IMP, we patched the Satellite IMP code to disable the reading at program start-up of the memory locations assigned to the configuration PROM. The distant host cable, having a round 31-pin Amphenol connector at one end and a flat 37-pin Cinch connector at the other end, was overlooked when the equipment was shipped; this item was sent by Federal Express and arrived after the BBN installation personnel had left the site.

Both the primary and alternate PSP terminal data interfaces (C/30 modem interfaces #3 and #4) worked successfully, as indicated by the Satellite IMP's receiving its own transmitted packets when the PSP terminal was set to the data loopback state. Both the primary and alternate CMM interfaces (C/30 modem interfaces #5 and #6) worked successfully, as indicated by the echoing of C/30 commands to the CMM and by the changing of the PSP terminal operation state. The Data Test Set was also used to verify correct operation of these commands.

Because we observed that an excessively long time was required for the C/30 Satellite IMP to achieve reservation synchronization, we concluded that a timing problem existed. Later, after the installation personnel had left the site, an interrupt handling bug was identified. When the bug was eliminated, synchronization in the C/30 Satellite IMP was indistinguishable from synchronization in the Honeywell 316 Satellite IMP.

In the beginning, communications to the COMSAT local network (DCN) could not be established, because the 1822 Host-to-IMP code in the Satellite IMP was incorrectly setting a timer to time out all input transfers after 15 seconds. As an interim measure, we patched the code to eliminate the time-out mechanism entirely. At a later date, DCN connectivity to the Clarksburg Satellite IMP failed again, possibly due to a hardware problem in the 1822 Host-to-IMP interface. We have yet to diagnose and fix the problem.

2.2 Hardware Maintenance Operations

During the last quarter, several hardware problems appeared which we diagnosed and, when they were related to the Satellite IMPs, fixed. A major problem marked by severe channel degradation causing several Satellite IMP crashes occurred in January, while the International Cooperation Board, the International Configuration Control Board, and the SATNET meetings were held at SRI International. Internet traffic from the gateways and ARPANET traffic on ARPANET Line #77 (the ARPANET direct connection via SATNET for London TIP connectivity) were interrupted repeatedly. As part of the diagnostic procedure, transmissions from the Etam and Tanum Satellite IMPs were inhibited to allow the Data Test Set attached to the Goonhilly PSP terminal unrestricted use of the channel. The cause of the problem was eventually traced to interference 8-10 dB above the noise floor on the SPADE satellite channel pilot signal, nominally generated by the Etam earth station only. Interference occurred first when the Greece earth station generated a signal about 100 Hz away from the pilot signal and later when the Mexico earth station generated a signal about 400 Hz away from the pilot signal.

Because of a reassignment of floor space at the Etam earth station, the COMSAT Earth Station Operations Center requested that the Satellite IMP and PSP terminal be relocated 30 feet away from their original locations. Accompanying the relocation were major changes in the associated earth station electronic equipment. Although repeated postponements of the relocation created havoc with the scheduling of BBNCC maintenance personnel, they were present during the move to inspect the Honeywell 316 computer.

Despite all precautions, the move did not go smoothly. Due to faulty cabling, large packet losses developed on the landline between the SDAC IMP and the Etam Satellite IMP and were not eliminated until the following day. Channel problems due to maladjusted frequencies and transmit power levels from the new channel equipment and from the PSP terminal hardware caused Etam to receive its own transmissions correctly but miss almost 80% of the European transmissions. Since the weekend Etam earth station personnel were unwilling to make major adjustments to the equipment, the channel degradation continued until the regular personnel augmented with COMSAT Laboratories personnel arrived the following week. Although the London TIP was isolated during the channel degradation, TCP connections could be established between UCL users and ISI machines, albeit with substantially poorer delay and throughput performance.

We have seen evidence of broken packets in reports to both the Catenet Monitoring and Control Center (CMCC) and the SATNET Network Operations Center (NOC). Monitoring data appear to be corrupted, even though the checksum seems to be valid; e.g., unrealistically large throughputs were being recorded in reports from the UCL and RSRE gateways, and unrealistically small throughputs were being recorded in reports from all the Satellite IMPs. No notable errors were seen on ARPANET Line #77, which passes through the Etam and Goonhilly Satellite IMPs but bypasses all the gateways. Our best guess at the moment is that legitimate packets are occasionally being corrupted by either the Etam Satellite IMP or the BBN gateway. There may be either bad locations in buffer memory or malfunctioning hardware on the VDH interface (before the checksum is calculated or after the checksum is checked). The recent move of the Honeywell 316 at the Etam earth station makes this machine a prime suspect. The major difficulty in diagnosing this problem is that we do not know how to make it appear consistently.

Other hardware problems occurring in SATNET are as follows. The Goonhilly Satellite IMP abruptly ceased receiving packets, causing a lengthy disruption to SATNET operations. Early on we determined that the Satellite IMP worked externally crosspatched (into the PSP terminal interface and immediately back to the Honeywell 316 interface). The problem was eventually traced to a malfunctioning modem in the B channel of the PSP terminal; to restore SATNET operations, we switched to the backup A channel. Subsequently, the modem was shipped to the United States for repairs. Two days later, severely degraded packet reception from Goonhilly again caused lengthy disruptions. This time the problem was traced to an intermittent in a connector on the A channel of the PSP terminal. Corrective action taken by site personnel was to remove the working modem from the A channel for installation into the B channel, which was subsequently enabled for site operations.

Modems in the 9.6 Kb/s circuit between the Goonhilly

Satellite IMP and the London TIP entered into an unknown state, creating a one-way circuit. Following usual procedures, the corrective action taken was for Goonhilly site personnel to loop and immediately thereafter unloop the modem. Another one-way circuit between the BBN gateway and the Etam Satellite IMP isolated the entire European internet community except for those users able to specify ARPANET Line #77. TELCO was called in and eventually traced the cause of the problem to a broken transmit wire in the cable between the Bell 303 modem and the BBN gateway.

What was supposed to have been a routine hardware repair of the Pluribus E bus at the SDAC IMP lasting only one hour turned into a serious problem, isolating all European traffic on ARPANET Line #77. After six hours of outage, the Pluribus E bus was repaired, and service was restored.

A power supply malfunction in the DEC PDP-11/40 serving as the BBN gateway isolated European internet traffic; repairs were effected by DEC maintenance personnel. After much effort and despair in trying to restore the circuit between the NDRE gateway and the Tanum Satellite IMP, Norwegian personnel discovered to their dismay that their power cable for the modem was missing. Unexplainably the Etam Satellite IMP was on one occasion powered off at the Honeywell 316 console. Because of an improperly set SPADE up-converter at Goonhilly, the Goonhilly Satellite IMP was inhibited from transmitting; earth station site personnel reset the up-converter to restore operation.

3 PLURIBUS SATELLITE IMP DEVELOPMENT

The major activity during the quarter continued to be the support of Wideband Network system integration. Early in the quarter substantial progress was made in bringing the many elements of the network together for a demonstration of cross-country packet speech. A plan to consolidate this progress in the form of regularly scheduled two-day-per-week availability of the network to experimenters was delayed by numerous equipment failures during the remainder of the quarter.

In November, the focus of all contractors was on the speech demonstration scheduled at mid-month. During the first week of November BBN personnel joined Linkabit personnel at ISI to check out a new, more robust version of the ESI delivered to that site. By 6 November it was possible to loop a voice conversation over the satellite channel at ISI for the first time. The Wideband Network demonstration on 18 November involved two simultaneous full-duplex 64 Kbps voice calls between the Lincoln Laboratory and ISI sites. At Lincoln, both calls used Packet Voice Terminals connected to the PSAT via a Lexnet and Miniconcentrator Gateway. At ISI, this same configuration was used for one call but the other call terminated in a speakerphone connected to the Wideband Network via the ISI-developed Switched Telephone Network Interface. The demonstration was carried out operating the channel at 772 Ksymbols/sec BPSK without coding. An attempt to run at the full 3 Mbps channel rate was unsuccessful due to an inability of the modems to lock onto one another. The demonstration was viewed as quite successful and a major milestone in spite of several problems which occurred due to the lack of robustness of the system in the presence of noise.

Multisite testing in December was essentially impossible because of the unavailability of the Lincoln ESI all month (from shortly after the November demo until early in January). Use of the DCEC site as a substitute was not practical due to a lack of speech equipment and a continuing problem with terrestrial access at that site. Progress in the month of December was largely in the area of host interface checkcut involving the Voice Funnel, Packet-to-Circuit Interface (PCI), and the Miniconcentrator Gateway. On 3 December, the first Voice Funnel was installed at Lincoln Laboratory and initial checkout with the PSAT was accomplished. On 17 December the first stream transmission between the Voice Funnel and PSAT was tested in support of single full-duplex voice conversation. Prior to this time all interactions between the PSAT and Voice Funnel had involved only datagram messages. On 10 December, the PCI was successfully tested with the PSAT for the first time. A telephone conversation over the satellite was initiated between a telephone connected to the PCI and a Lexnet Voice Terminal.

Miniconcentrator Gateway software was modified in December to support stream creation/deletion without manual intervention. Checkout of this feature with the PSAT was also accomplished. During December, the PSAT at DCEC experienced a hardware outage that was traced to a bad memory card and replaced.

The inability to support multisite experimentation which has been described above for December persisted throughout January. During the first half of the month (until January 11), the Lincoln site was essentially unavailable because of the ESI problem already noted. During the second half of the month, the ISI site was plagued by degraded performance on the channel, the exact cause of which is still under investigation at the time of this writing. A number of Western Union equipment failures and environmental difficulties at the Lincoln and ISI sites were identified and corrected. Some of ISI's difficulty can be blamed on the old RFI problem which was again observed by Linkabit personnel on several occasions. From 5 January through the end of the month, the DCEC site was limited to degraded performance due to operation with the spare 75 watt HPA. On 20 January a SuperSUE poller was installed at DCEC in preparation for the installation of a PCI in early February. Tests with the site on 22 January indicated an inability to operate the ESI in any of its loopback modes, thus precluding further testing at that site until the problem was corrected.

Some progress in host-to-PSAT interfacing was accomplished, however, in January. A bug in the PSAT burst aggregation software had for some time prevented operation of the Voice Funnel in stream mode over the satellite channel. In mid-January this bug was traced to an unexpected interaction between the datagram and stream mechanisms and was corrected. On 16 January the Voice Funnel at Lincoln Laboratory looped a speech call over the satellite channel for the first time. On 28 January the first voice call between Voice Terminals on the Voice Funnel and the Miniconcentrator Gateway was established. Cross-country tests between the Lincoln Voice Funnel and ISI Miniconcentrator Gateway were held up waiting for the performance of the ISI site to improve. During the period 19 - 27 January, tests involving the PCI were carried out over the satellite channel at Lincoln Laboratory in preparation for shipment of the unit to DCEC. On 26 January BBN verified that the PSAT was able to control the two Earth Terminal loopback modes and monitor Earth Terminal status via the ESI at ISI. Modifications which will allow this same capability at the other two operational sites is scheduled in the near future. It is important to note that while many of the activities described above do not directly relate to the PSAT, BBN personnel have spent many hours working with site personnel and the other contractors supporting integration tests and diagnosing network and subsystem failures.

In an attempt to improve the communication between Wideband Network contractors and thus facilitate the network integration process, BBN has been producing a Biweekly Status Message describing progress and problems which have occurred. In January the format of this message was expanded in order to provide more quantitative and graphical information on the operational status of individual sites and the aggregate use of the shared satellite channel.

During the quarter, work necessary to convert Wideband Network monitoring software (which runs under the TENEX operating system) to the NU system (under UNIX) continued. In addition, up until the end of the quarter work continued on conversion of the PSAT application software and operating system for compatibility with the new C/70 Pluribus assembler/linking loader. By the end of January, however, it had become apparent that this latter conversion task was substantially larger than we had originally anticipated. Because of our limited personnel resources and the many higher priority tasks yet to be accomplished, we decided to (at least) temporarily defer continuing this assembler conversion activity. The primary disadvantage of this decision is that it will continue to be relatively costly and inconvenient to reassemble the PSAT program and thus potentially slow down our ability to add new features.

Another activity carried out during the quarter was the development of a preliminary design for a stream synchronization algorithm for the PSAT. Both active schemes, where a crashed site requests the stream database from one of the in-sync sites, and passive schemes, where stream database information is continuously transmitted within the stream subframe, were considered. The basic approach decided upon is similar to the one currently used by SATNET. A more detailed discussion of this approach will be contained in our next Quarterly Technical Report.

Our proposed option to develop a Butterfly-based PSAT (BSAT) was approved in December; approval led to more detailed interactions among the members of the PSAT group and the Butterfly hardware and software developers beginning in January. Our current intention is to maintain the current PSAT structure to the extent that it can be mapped efficiently onto the Butterfly hardware and (Chrysalis) operating system. Ways of temporarily decoupling the BSAT development from any possible redefinition of the PSAT/ESI interface were also explored during the quarter.

Finally, during the quarter the details of the new Wideband Network addressing scheme as they relate to the PSAT were worked out and the software to implement these modifications was written. The details of this scheme are described in section 3.1 below.

3.1 New PSAT Addressing

The basic Wideband Network addressing scheme that was adopted at the Wideband meeting at Lincoln Laboratory in May 1981 is summarized in W-Note-27. That note describes the format of the Internet and Wideband Network addresses, and the mapping between them. The following discussion expands upon the structure and interpretation of the Wideband Network (HAP) address. Also included below is a list of assigned Wideband Network addresses, excluding addresses which are sub-addresses of a "real" host (primarily Lexnet terminals and Voice Funnel internal hosts).

3.1.1 Host Address Types

The basic form of a Wideband Network address is a 16-bit number, divided into two or three fields depending on address type. The four basic types of addresses are described below.

(1) REAL HOSTS are hosts associated with PSAT interface devices. This includes all hosts attached to PSATs (Miniconcentrators,

Voice Funnels, etc.). Gateways, whether internal or external to the PSAT, are also members of this class. This allows for changes between internal and external gateways without disturbing the address of the gateway. The address format for real hosts is the one described in W-Note-27, namely:

```
15             8 7             0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   HAP Number  |     Local     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The HAP address is split into two fields, each 8 bits long. The most significant byte is the so-called "HAP Number" field, which is used by the PSAT for routing. The lower byte is the "local" field, which is passed along by the PSAT, but ignored for PSAT routing purposes. The PSAT allows multiple HAP numbers to refer to the same host (aliasing). The following HAP numbers are currently assigned:

08	ISI PSAT Internal Gateway
09	Lincoln PSAT Internal Gateway
0A	DCEC PSAT Internal Gateway
0B	SRI PSAT Internal Gateway
0C-0F	(Reserved for Future Gateways)
10	ISI Miniconcentrator
11	ISI Voice Funnel
12	Lincoln Miniconcentrator
13	Lincoln Miniconcentrator #2 / PCI
14	Lincoln Voice Funnel
15	DCEC Packet-to-Circuit Interface
16	DCEC Voice Funnel

17	SRI Miniconcentrator
18	SRI Voice Funnel

(2) INTERNAL HOSTS are hosts which are software processes internal to each PSAT, but have distinct Wideband Network addresses. These hosts are used for host and network testing, resource management, and network monitoring and control. Real hosts which want to use the facilities of internal hosts will usually want to address the corresponding Global Host Address (described below). The global host will get service from the appropriate internal host in the local PSAT, and can be accessed through a convenient network-wide name. More detailed descriptions of the internal hosts can be found in section 3.1.2 below. The format of internal host addresses and the identity of those internal hosts currently defined are as follows:

```
15       11 10          4 3     0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0| PSAT Site # | Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type	Internal Host	Comments
0	Service	
1	Query	Planned
2	Message Generator/Sink	
3	Echo	
4	Speech Simulator/Sink	
5-7	Unassigned	
8	Network Monitoring	BBN Use Only
9	EXPAK Server	BBN Use Only
A-D	Unassigned	
E	Packet Core Server	BBN Use Only
F	PSAT Control	BBN Use Only

The PSAT site number is an integer in the range 1-127 decimal which is different for every node. The following site numbers are currently assigned: ISI=1, Lincoln=2, DCEC=3, SRI=4, and RADC=5. No site will ever have the site number 0. Internal hosts marked "BBN Use Only" are control hosts and should not have messages sent to them by user hosts. However, they are moderately well protected in case they should receive some unintended messages.

(3) GLOBAL HOSTS are addresses which are always local to the subscriber's node. These hosts are generally used for host testing and network management. If a message with a global host as a destination address is sent over the satellite channel, the

appropriate host at all PSATs will receive the message (essentially an implicit group address). However, hosts will not normally have access to this particular feature of global addresses (nor should they need it). Global hosts are intended to be aliases for regular PSAT internal hosts, but with network-wide constant addresses, so that the real hosts at a node need not know which node they are attached to. The most important global host is address 0, the local service host, which is used for stream and group management functions. The address format and identity of current and planned global hosts are as follows:

```
 15       11 10          4 3     0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0|0 0 0 0 0 0 0|N N N N|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address(NNNN)	Name
0000	Service host
0001	Query host
0002	Message Sink
0003	Local Echo host
0004	Satellite Echo host
0005-000E	Unassigned
000F	PSAT Control host, BBN use only

(4) GROUPS are dynamically created network addresses for collections of hosts. All messages sent to a group address are sent over the satellite channel, regardless of the number and location of group members. The format of group addresses is:

```
15        11 10                      0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1|          #           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where '#' is the group number. Groups addresses are thus in the range F800-FFFF.

3.1.2 Internal Host Descriptions

Internal hosts in the PSAT fall into two classes: user and control. User internal hosts provide services to Wideband Network users, while control internal hosts are reserved for BBN use only. Documentation on control internal hosts is internal to BBN, while user internal host services are described here or in other published documents. Users should not send messages to control hosts, although these hosts are protected against unwanted traffic. Internal (and Global) hosts are:

SERVICE provides user hosts with network resource management functions. The service host is fully documented in Section 4.1.4 (Setup Level Messages) of BBN Report No. 4469 (PSAT Technical Report). The service host should always be addressed through its global address (0000), rather than by its internal host address. This ensures that the local PSAT's service host is the one that receives the message. Strange behavior is likely to result if a host sends a request to a non-local service host.

QUERY is as yet undefined, but has been reserved since it appears to be an internal host that will be useful someday. It would probably provide users with information on the state of network nodes and hosts. No work on this host is planned for the near term.

MESSAGE SINK is nothing more than a bit-bucket. Messages sent here are counted by the PSAT, and certain other statistics can be collected by BBN for traffic sent here. The sink has a global alias, which refers to the local PSAT's message sink, primarily to aid in host testing. There is also a message generator associated with the sink. The message generator is not directly accessible to user hosts, but experimenters may control it using facilities provided by BBN (the EXPAK program).

ECHO provides a datagram echo service at the HAP and Internet levels. This host functions by swapping the source and destination HAP address fields of a received message, and sending the resulting message. If the HAP type-of-service word indicates that the message is Internet, then the IP source and destination address will also be interchanged. Stream traffic should not be addressed to the echo host, due to symmetry problems in the HAP header. If a stream message does reach the echo host, it will be discarded. Datagrams addressed to an echo host in a PSAT other than the local node will transit the satellite channel twice, for a round trip delay of about 1.2 seconds. Datagrams addressed to the local PSAT's echo host are locally delivered as fast as processing allows. The echo host has two global aliases. The "Local Echo Host" is an alias for the local PSAT's echo host, which guarantees local delivery. The "Satellite Echo Host" is also an alias for the local PSAT's echo host, but the reflected datagram is forced to be delivered over the satellite channel. This feature supersedes the undocumented "Force Channel Transmission" bit in the HAP message control word. The round trip path of a datagram in this case transits the channel once, for a round trip delay of about 600ms.

SPEECH GENERATOR/SINK is a special-purpose message generator/sink for making measurements of network behavior with simulated speech traffic. This sink should not be used by other hosts, as it is designed specifically to make traffic measurements on simulated speech traffic. Further documentation on this host, as well as on the MESSAGE GENERATOR/SINK, can be found in W-Note 8.

NETWORK MONITORING is used by BBN to collect periodic status and throughput information on the network and its hosts. This host should not be addressed by users. The information provided by this host can be provided by BBN to users in the form of daily network summaries. Access to a program for monitoring the network in real-time is also possible.

EXPAK SERVER is a server for a special network control program for changing certain program parameters in the PSAT. These parameters include those controlling the PSAT's message generators and service host. Users can obtain access to this control program (EXPAK) from BBN on request, but should not address messages to this host directly.

PACKET CORE SERVER is used for changing and examing the PSAT's memory image remotely over the Internet. Users should not send messages to this host, nor is access to the related control program permitted to network users.

PSAT CONTROL is a host used exclusively by BBN for certain network management fuctions, notably stream synchronization. Users should not address messages to this host.

3.1.3 Expandability

The formats described above should allow a reasonable amount of room for future expansion of the Wideband Network. The allocation of HAP number space is as follows:

```
     +----------------------------------------+
  FF ¦                                        ¦
  .. =           2048 Group Addresses         =
  F8 ¦                                        ¦
     +----------------------------------------+
  F7 ¦                                        ¦
  .. =             232 Real Hosts             =
  10 ¦                                        ¦
     +----------------------------------------+
  0F ¦                                        ¦
  .. =     8 Internal or External Gateways    =
  08 ¦                                        ¦
     +----------------------------------------+
  07 ¦                                        ¦
  .. =     16 Global Hosts, 127 PSATs         =
  00 ¦       with 16 Internal hosts           ¦
     +----------------------------------------+
```

There are 256 possible HAP numbers. With 8 HAP numbers (00-07) reserved for global hosts and PSAT internal hosts, and 8 HAP numbers (F8-FF) reserved for group addresses, there are 240 HAP numbers remaining for real hosts (08-F7). Eight of these numbers (08-0F) are allocated to gateways, or reserved for future gateways, but this assignment is for mnemonic convenience only, not a restriction of the format.

The block of HAP numbers reserved for group addresses allows for 2048 groups active in the net at any one time, which should prove more than adequate. However, real hosts will be allocated addresses starting from the low end of the address space, allowing for expansion of the group address space if this becomes necessary.

The reserved space for PSAT global addresses and internal hosts allows for 16 global addresses with 1 currently assigned, 5 assigned to envisioned new hosts, and 10 unassigned. The format allows for 127 PSAT nodes each with 16 internal hosts. There are currently 4 PSAT nodes built, with a few more under construction. Each PSAT currently has 7 internal hosts, and 2 more are envisioned in the near future, leaving 7 unassigned.

4 REMOTE SITE MAINTENANCE

The heart of remote maintenance is software control and distribution. If working software is distributed correctly, there will be fewer problems requiring the attention of the system maintainers, allowing them to devote their attention to genuine software bugs. In an environment where software is continually being developed and improved, and where bugs are being fixed routinely, considerable effort is required to keep the software at remote sites current with that of the development site. Various ad hoc schemes, including site visits, have kept the problem at bay in the ACCAT Remote Site Modules. Not only are such schemes expensive, since they use a fair amount of human resources, but they are prone to error, resulting in confusion and inconvenience while these installation errors are detected and repaired. During the end of the last quarter we began serious development of a program intended to automate software distribution, with the desire to eliminate software outages due to mere error in installation.

The RSM system software development is concentrated in a single machine. In the discussion below, this machine, which receives frequent installations of new software, will be called the source machine. In the simplest case, the other machines, called the target machines in this discussion, would be running software which is identical to that installed on the source machine. A more realistic distribution system will allow for exceptions. The goal is to distribute new software to the targets and to reflect any changes in file attributes (links, ownership, or protection) which occur on the source system.

The purpose of the Update program is twofold:

1) to determine which files that have changed on the source system since the last distribution are ready for distribution, and to effect the updates; and

2) to identify exceptions--i.e., programs on the target machines that SHOULD be different from their counterparts on the source machine.

This means that not only are the updates themselves automated, but easily forgotten details, such as the exceptions, become a matter of record, rather than a matter of lore.

The changes are driven by three databases: the Unix file-systems of the source and target machines, and a special database (described below) which describes each controlled directory for each machine.

The file-system on the source machine serves as a prototype working file-system, implicitly embodying the inter-relationships among the software modules. The software update mechanism tries to transform the target's file-system into a copy of this prototype. A status file is created for each controlled directory on the source. This file contains ownership, protection, and link information for all files in the directory (similar to a directory listing) as well as a checksum of the contents of these files. A similar file is created on the target machine, and retrieved to the source. The status files are compared, and where the entries in these files differ, the software update program decides what action, if any, is appropriate to rectify this difference. Commands to adjust protections, ownerships, or links, are issued to the target machine; if the checksums differ, a new copy of the file in question is sent, with instructions for its installation using the BBN-UNIX "install" facility.

It is not always appropriate for a remote system to slavishly imitate the source system. Hardware capabilities differ from machine to machine, requiring programs either to determine what capabilities are present at execution time or to be tailored for the individual site. Also, user habits differ, due to historical inconsistencies, or to differences in the purposes of each site. There are many places where these differences in habit or purpose affect software. For these reasons and others the UNIX file-system on the development machine does not provide enough information to maintain remote sites.

The software update program will decide which directories on the target are controlled either by being told explicitly by the user at execution time, or by reading a database file, similar to a Makefile. This database file will specify, for each directory:

a) Whether or not to remove files that appear in the target machine's directory and do not appear in the source machine's directory. Examples of directories in which unshared files are expected include /usr/lib and users' personal directories.

b) Whether or not to descend into and process subdirectories of a given directory.

c) How long files in this directory should be "cured" on the development machine before being distributed. Controlled software will normally be run on the source machine for some time before being distributed to remote sites.

d) Subdirectories or files in a controlled directory that should not be be updated (exceptions).

e) A preprocessing command to be executed in the directory on the source machine (an example: "make rmobjects").

f) A preprocessing command to be executed on the destination machine before doing the update.

g) A post-processing command to be executed on the source machine after the update has taken place.

h) A post-processing command to be executed on the target machine (e.g., "make all.install").

Machines of different architectures (or even different operating systems) can be maintained with this mechanism by controlling source directories rather than binary directories. Indeed, it is not even necessary for the architectures of the machines to differ. For the most part, the CINCPACFLT and Naval Postgraduate School machines can be updated by maintaining their directories of binaries, /bin, /usr/bin, etc., with the sources being kept on the machines as a matter of courtesy. The NOSC machine, for historical and hardware reasons, will be maintained by updating source directories, with a "make all.install" post-processing command, to allow for its different disk structure, and for the peculiarities of its terminal handler defaults.

The software update uses the command execution capabilities we recently added to the FTP program. It interrogates the target about the contents of controlled directories, and updates these directories where necessary. In most situations, transactions between the source and target machines will take place through an FTP connection. However, it is occasionally convenient to perform an update by copying the new files from a mounted file-system on disk to the controlled directories on the system. For this reason, the transfer mechanism is designed to support such transfers, so that the control structure is the same for an FTP update or a disk-to-disk update, and so that only at a low level is the difference in the transfer mechanisms apparent.

We present here a rather detailed Pidgin-Algol explanation of the algorithm to be employed.

```
PROGRAM UPDATE
BEGIN
    /* Parse the arguments. */
    getargs(argc, argv);
    InterceptSignals();

    /* remember invocation time as InvokedAt. */
    time(&InvokedAt);

    /* broadcast a warning to users not to change the
       affected directories on the target machine during
       the update and also append message to /etc/motd
     */
    if(SrcWarningRequired)
        WarnSrcUsers();

    /* First loop: parse the database, get the login
        information etc., for each host to be done.
        The list of hosts is obtained either from the
        invocation  arguments or from a file, e.g.,
        /usr/lib/update/default.hosts.
     */
    FOR All_Hosts DO
    BEGIN
        /* the parser is a YACC parser based on the one
           used in MAKE to parse Makefiles.
         */
```

```
Parse the host database;

/* the following obtains a valid password for the
   current host.
 */
Ask User for update password on the host currently
    under consideration
IF the validate-password flag is set
          AND IF this is a network update THEN
BEGIN
    IF openning the ftp connection to
        the current host succeeds THEN
    BEGIN
        WHILE the password remains unvalidated DO
        BEGIN
            log the user in.
            IF the login fails THEN
                ask for the password again.
            ELSE
            BEGIN
                Note that the password for
                    this host has been validated.
                close the ftp connection.
            END
        END
    END
    ELSE BEGIN
        Since you can't open an FTP connection to
            the host, report that you aren't able
            to verify at this time.  Give the user
            three choices:
            forget all about updating this host,
                i.e., delete it from the linked
                list of hosts to do
            try again now (only useful for local
                transfers in which the user is
                able to remedy the cause for
                the failure to connect, e.g.,
                by rebooting the system).
            hope that the host is up and the
                password works when you get to it
                (i.e., skip the verification step
                for this host).
    END
END
Semi-encrypt the password.
/* This program will be sitting on the system for
    some time, possibly for several hours, with
```

```
                the password strings held in memory.  There
                is a problem here: we can't completely encrypt
                the password because we have to decrypt it
                later on, to submit it to the other host.  No
                matter how clever you are about hiding the
                strings, the algorithm for retrieving them
                will be sitting in core for anyone who can
                read /dev/kmem to examine.
               The alternatives are:
                - Change the kernel to permit special "trusted"
                    connections, then perform these transac
                    tions through that kind of a connection.
                    This technique eliminates the need for
                    this program to know the user's password.
                    This would be a nice thing to do in the
                    current RSM environment, but it is not very
                    portable.  The mechanism as presented here
                    is portable to any system which has the XCMD
                    protocol implemented.  FTP protocols are
                    much easier to implement (e.g., on TENEX)
                    than special trusted connections.
                - Open all the connections now, log the user
                    in, and forget the passwords. This ties up
                    network resources, and is also unreliable
                    as hosts are prone to crashing and network
                    connections are prone to closing without
                    warning.
                - Keep all connections open but do everything
                    in parallel and in a distributed manner.
                    This is a nice solution, but again ties up
                    many network resources.
          */
    END

    /* This is the loop where the action takes place.

       "all hosts": chase down the doubly-linked
           list of host structures.

     */
    FOR all hosts DO
    BEGIN
        IF this is a disk to disk update
            OR openning an ftp connection to
            the host succeeds THEN
        BEGIN
            IF this is a disk to disk update
                OR logging in as whoever succeeds THEN
            BEGIN
                FOR all directories/files or groups to be
                    updated DO
                    Call DirectoryProcessor with the
                        directory structure as an argument

            END
            ELSE
                Since you can't log in, complain that the
                host is not accessible at this time.

        END
        ELSE BEGIN
            /* If the host is down now, it may be up later
               in the evening.  We will try the host
               again later, unless our patience is
               already exhausted.
             */
            IF hostp->h_attempts < maxtries THEN
            BEGIN
                increment the number of attempts on this host.
                move the host structure to the end of the
                    linked list of host structures to be done
            END
            ELSE
                Since we've tried, and failed to connect to
                    this machine several time, report that
                    this host is a total loss.
        END
    END

    /* remove the /etc/motd entry saying there is an
       update in progress, perhaps broadcast a message
     */
    UnwarnUsers();
    remove the status files
END PROGRAM SoftUpdate



PROCEDURE DirectoryProcessor( a directory/group structure )
BEGIN

    /* start by issuing the pre-processing commands on the
       source and target machines.  A failure of either of
       the commands to execute is considered fatal for
       this directory, and causes this subroutine to return.
     */
```

```
IF this directory/file/group has a
    Source pre-processing command defined, THEN
BEGIN
    issue the command.
    IF command has non-zero exit status THEN
        log the failure, abort from procedure
END
IF this directory/file/group has a
                   Target pre-processing command THEN
BEGIN
    issue the command on the target.
    IF command has non-zero exit status THEN
        log the failure , abort from procedure
END

IF this directory is marked for processing
    subdirectories THEN
BEGIN
    FOR all subdirectories in this directory DO
        IF this subdirectory is not an exception
        THEN
        BEGIN
            Allocate a new directory structure.
            Copy the current directory structure into
            the new one, except:
                The name of the entry is the absolute
                 pathname of the subdirectory
                df_destname is the absolute pathname
                    of the destination subdirectory
                    (constructed by appending this
                    subdirectory's name (on the source
                    machine) to the name of the parent
                    directory on the target machine).
                df_exceptions has only exceptions
                    pertinent to that subdirectory.
            Call the DirectoryProcessor with this
                subdirectory structure as argument
        END
END

Run the program Update_status on the other machine
Get the output of Update_status to the local machine.
/* We make a file "Source.directory" for this direc-
   tory we are working on.  Since we will probably be
    updating this directory onto several hosts in this
    incarnation of this program, we can just build the
    Source.directory file once.  The following com-
    plicated check is a means of insuring that the
```

```
        existing Source.directory file is up-to-date
        and therefore doesn't need to be re-created.
    */
    IF the file Source.directory doesn't exist OR
      IF the file Source.directory is older than InvokedAt
        THEN
            Run the program Update_status on this directory
               on the source

    Read in and parse the two Update_status files, deter-
      mining the differences:
        Make sure multiply-linked files have the same
            links on both machines
        Make sure xsums are the same
        Make sure the ownerships are the same on both
            machines.
        Make sure the group-memberships are the same on
            both machines.

    Construct the list of unshared files.
    FOR all files in Target.directory whose data differs
        from the data in the Source.directory
    BEGIN
        IF the file is not an exception (in the database
            file) THEN
        BEGIN
            /* We want to distribute files that have been
               "cured" on the system for a minimum number
               of days, determined on a per-directory,
               per-host basis.
             */
            do a stat on the file on the source machine to
                determine age one might argue that this
                check is redundant, however, the file
                might have changed while this program is
                running, so checking now insures that you
                won't be installing a program that hasn't
                een cured.
            IF the Source file is not old enough
                           AND it is a regular file THEN
            BEGIN
                /* Putting this check here, rather than
                   checking the age of the file when con-
                   structing the list of files to be
                   shipped, has an interesting (and desir-
                   able) side-effect: if some zealous
                   programmer has installed a new file on
                   the Target host within the age window,
              then it will be identical to the one
              on the source machine, and thus would
              have been eliminated from consideration
              when the two Update_status files are
              compared.
            */
           Find the correct Source file
           Use the source file obtained, instead of
            the too young one.
           Get the information (size and checksum
            only) for this file.
           /* If the backup file matches the file on
               the target machine there is no need
               to perform an update of this file.
              Early versions of backup did not preserve
               ownership, modes, etc.  Therefore we
               use the modes of the version currently
               installed.  This option is not perfect,
               but it should solve more problems than
               it causes.
            */
           IF the information of the backup file
               matches the info of the file on the
               target machine THEN
                   continue to the next file to be
                       updated
       END

       IF file types are different or special file
           missing THEN
               log problem.
       ELSE IF xsums are different THEN
       BEGIN
           /* transfer, check, install */
           IF this is a network (ftp) update THEN
           BEGIN
               WHILE attempts < three
                   AND store not succeeded DO
               BEGIN
                   store the file from Source to Host's
                       staging area
                   IF the checksum of the stored file
                       is correct THEN
                           the store succeeded
                   ELSE
                       increment attempts
               END
               IF store not succeeded THEN
```

```
                        IF doing group THEN
                        BEGIN
                            abort group
                            /* send cmd to remove any group
                               temp files already there
                             */
                        END
                        ELSE
                            continue and do next file
                        END
                    END
                END
                ELSE
                    /* the store did succeed, or this is
                        disk-to-disk.
                      */
                     IF doing a member of a group THEN
                        add proper command to install
                        string
                        /* the install string is fed to the
                           install cmd on the destination
                           machine. It consists of lines
                           like "install file dir flags
                           -and file dir flags ..."  All
                           the members of a group are
                           installed at once.
                          */
                     ELSE   /* doing an ordinary file */
                        install the file in the correct
                            place.
                     END
            END
            ELSE
                 /* don't have to store, the file, just
                    issue a command to change modes...
                  */
            DO
            BEGIN
                IF the difference is ownership
                    THEN change the ownership
                IF the difference is group
                    THEN change the group
                IF the difference is protection
                    THEN change the protection.
            END
        END
    END
    /* we are now finished updating the files for this
       directory...
     */
    IF this directory has the "Remove unshared files"
        flag set THEN
        FOR all files on Host that aren't on Source
            delete the file from Host
    IF this directory has a Source post-command THEN
    BEGIN
        issue the Source post command
        IF command has non-zero exit status THEN
            log failure and status
    END
    IF this directory has a Target post-command THEN
    BEGIN
        issue the Target post-command
        IF command has non-zero exit status THEN
            log failure and status
    END
END
```

5 INTERNET DEVELOPMENT

The major activity during the past quarter was the development and deployment of the Macro-11-based PDP-11 gateways. Other work included continuing operation and maintenance of the BCPL and Macro-11 gateways, supporting the new Internet Control Message Protocol (ICMP) in the Macro-11 gateway, supporting class B and C internet addresses, monitoring the gateways from the Network Operations Center (NOC), measuring the performance of the Macro-11 gateways vs. the BCPL gateways, and ordering a 56kb host line from Telenet for the VAN gateway work.

5.1 Macro-11 Gateway Description

The Macro-11 gateway is written in PDP Macro-11 assembly language. It runs under the MOS operating system, in either LSI-11/03 or PDP-11/40 processors; we also plan to run it in an LSI-11/23 when one is made available for testing.

Although the Macro-11 gateway is quite similar in functionality to the BCPL gateway which it replaces, several important changes have been incorporated. These are as follows:

- The Macro-11 gateway uses considerably less code space than the BCPL gateway, currently 13k words vs. 22k. This leaves considerably more memory available for buffers and for installation of new features.

- The gateway has improved strategies for utilizing network resources. One example of this is that for ARPANET-like networks it counts RFMNs to prevent the IMP from blocking the gateway.

- The gateway's buffer management is more tailored to its network usage. When a message is read from an interface, the gateway tries to put up another buffer from the free buffer pool. If none is available and if the packet is a data packet, it is reused for input. If the packet is a control packet (e.g., GGP routing update) it will be processed and then reused. This strategy has two important advantages. First, by always being willing to accept messages the IMP will not declare the gateway tardy; second, when the gateway is running low on buffers, control packets are processed ahead of data packets.

- The gateway supports XNET protocol to a running gateway. In contrast the BCPL gateway would halt when someone tried to XNET to it. The Macro-11 gateway allows us to examine and change it while it is running. The usefulness of this feature when diagnosing problems cannot be overstated.

- The operator interface in the gateway has been improved. There are commands to print out information about the gateway's neighbors, its interfaces, and the other nets it knows about, put this into its loader, restart it, and start DDT; and there is a help command which explains the other commands.

Because the Macro-11 gateway has much more free memory than the BCPL gateway, many new features have been added. The new features are as follows:

- The new Internet Control Message protocol (ICMP) is supported in the Macro-11 gateway. This will replace the part of the Gateway-Gateway protocol (GGP) that provided host services. ICMP has been implemented is such a way that it can be enabled by setting a software switch, permitting the GGP to ICMP change-over to be done very easily.

- A message generator has been added to the gateway. This provides the facility to generate traffic in order to allow new code testing and problem diagnosis.

- The gateway fragments packets with options.

- The SATNET support has been improved to allow communication with a Satellite IMP running in its loader. Previously, when the Satellite IMP was in its loader the gateway would declare the interface down and not forward any packets. The Satellite IMP now sends packets to the gateway which have a bit set to indicate that it is in its loader. The gateway detects this and does not take down the interface. This permits the Satellite IMP to be loaded via the gateway.

- Support has been added for a V2LNI local network. This is installed in the NDRE gateway and a test gateway at BBN.

- The gateways are being monitored using the Host Monitoring Protocol (IEN-197) from the Network Operations Center (NOC). This is currently limited to Trap reporting and manual status commands. This will later be expanded to complete status, topology, and throughput support.

5.2 Macro-11 Gateway Installations

The Macro-11 gateway was installed at four sites. These are shown in the table below.

Gateway	Adjoining Networks	Installation Date
RCC	ARPANET - RCCnet	1/7/82
BBN	SATNET - ARPANET	1/7/82
NDRE	SATNET - NDRE/Ring - NDRE/TIU	1/15/82
UCL	SATNET - UCLnet - RSRE	2/16/82

5.3 Gateway Routing

We are in the process of implementing support for Class B and Class C internet addresses. In addition to having the gateway simply understand the format of the addresses, a new type of GGP routing update was needed to be exchanged between the gateways. To allow as many nets as possible to fit in one message we decided on the following format. The networks are arranged by distance, grouped according to what networks are at a given distance. The update is a new GGP message type (12) to prevent conflicts with the existing GGP routing updates. The format and a description of the fields in the message are as follows:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!Gateway Type=12!  unused (0)   !                 ; 2 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!     Sequence Number           !                 ; 2 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  need-update  !  n-distances  !                 ; 2 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  distance 1   !   n1-dist     !                 ; 2 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   net11       !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ; 1, 2 or 3 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   net12       !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ; 1, 2 or 3 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                .
                                .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   net1n1      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  ; n1 nets at dist 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                .                       ...
                                .                  ; ndist groups of
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                  ; nets
!  distance n   !   nn-dist     !                  ; 2 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   netn1       !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  ; 1, 2 or 3 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   netn2       !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  ; 1, 2 or 3 bytes
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                           .
                           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   netnnn      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  ; nn nets at dist n
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Gateway Type	12 (decimal).
Sequence Number	The 16-bit sequence number used to identify routing updates.
need-update	An 8-bit field. This byte is set to 1 if the source gateway requests a routing update from the destination gateway, and set to 0 if not.
n-distances	An 8-bit field. The number of distance-groups reported in this update. Each distance-group consists of a distance value

	and a number of nets, followed by the actual net numbers which are reachable at that distance. Not all distances need be reported.
distance 1	Hop count (or other distance measure) which applies to this distance-group.
n1-dist	Number of nets which are reported in this distance-group.
net11	1, 2, or 3 bytes for the first net at distance "distance 1".
net12	Second net.
...	
net1n1	Etc.

The new Class B and C routing update is being implemented in such a way as not to disrupt the existing routing. When a gateway comes up with the new Class B and C routing capability it will start sending the old and new routing updates to all of its neighbor gateways. When it gets an acknowledgement of a new routing update from a neighbor, it will stop sending the old format updates to that neighbor. This will permit switching over to the new format updates in an orderly way.

The handling of non-routing gateways in the Macro-11 gateway is being done in a slightly different way from previous versions. It uses the following procedures:

(1) Nets that are behind a Non-routing Gateway (NRG) will be known _a priori_ to Routing Gateways (RG). There can be one or

more of these nets. They will be thought of as directly connected to the NRG.

(2) The RG will forward traffic to the DG if they get datagrams addressed to the nets behind the DG. They will not send a Redirect to the source.

(3) The RG will not know of any other gateways (RG or NRG) behind an NRG.

(4) The RG will always use an RG as a path instead of an NRG if both exist. The NRG path will only be used if the RG path is down. When the RG path comes back up, it will be used again.

5.4 Monitoring

The Macro-11 gateways are now being monitored by the BBN Network Operations Center (NOC). Both the gateways and the NU monitoring host now support the Host Monitoring protocol (IEN-197). Currently there are two types of data available, namely, Traps and Status reports. The Traps are one-time events which are reported to the NOC. Examples of this are an ICMP Redirect being sent or a packet discarded due to a queue being full. An example of some trap output from the gateways follows:

```
Time  Name # Trap #     Description
----- ---- - ---- ----- -----------------------------------------

11:37 BBN  2 Trap 1000: RFNM count:pkt flshd 2 / 72
11:45 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 4.0.0.61 to 10.3.0.40
11:46 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 35.7.0.0 to 10.3.0.40
11:46 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 4.0.0.60 to 10.3.0.40
12:47 BBN  2 Trap 1000: RFNM count:pkt flshd 2 / 22
12:56 RCC  1 Trap 1010: ICMP RDR 10.1.0.5 -> 11.3.0.42 to 10.3.0.40
13:22 UCL  3 Trap 1010: ICMP RDR 4.0.0.64 -> 11.3.2.42 to 4.0.0.61
13:22 UCL  3 Trap 1012: ICMP type 5 from 4.0.0.64
13:22 UCL  3 Trap 1010: ICMP RDR 4.0.0.64 -> 11.3.2.42 to 4.0.0.61
13:22 UCL  3 Trap 1002: ENQ failed:pkt flshd 11.3.0.42 -> 11.3.0.42
13:27 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 35.7.0.0 to 10.3.0.40
13:27 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 4.0.0.60 to 10.3.0.40
13:28 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 35.7.0.0 to 10.3.0.40
13:28 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 11.3.0.42 to 10.3.0.40
13:31 RCC  1 Trap 1010: ICMP RDR 10.0.0.72 -> 4.0.0.60 to 10.3.0.40
```

The traps are indispensable for understanding and detecting problems in the gateways. Events which previously had gone unnoticed are now reported.

The other type of data collected from NU is status. There is a manual command that requests a gateway (or all gateways) to report their status. The gateways will report their version number, the status of their interfaces, the status of their neighbor gateways, and the networks they know about. An example of the output of the status commands is included below. It is the status of the RCC gateway which is between the ARPANET and the RCCnet (alias Div5net).

```
Gateway RCC 3.3.0.8 (Div5Net 3/8)

Version 1000

Interfaces:
   UP:     RCC 10.3.0.72 (Arpanet 3/72)
   UP:     RCC 3.3.0.8 (Div5Net 3/8)

Neighbors:
   UP:     DCEC 10.3.0.20 (Arpanet 3/20)
   DOWN:   BRAGG 10.0.0.38 (Arpanet 0/38)
   DOWN:   BBN 10.3.0.40 (Arpanet 3/40)
   UP:     C3PO 10.1.0.51 (Arpanet 1/51)
   UP:     R2D2 10.3.0.51 (Arpanet 3/51)
   UP:     MIT-LCS 10.0.0.77 (Arpanet 0/77)
   UP:     PTIP 3.2.0.5 (Div5Net 2/5)
   UP:     PTIP 10.2.0.5 (Arpanet 2/5)
   UP:     FIBERNET 3.2.0.50 (Div5Net 2/50)
   DOWN:   MILLS 10.3.0.17 (Arpanet 3/17)

Network Table:
   net 10  Directly connected
   net 3   Directly connected
   net 24  2 hops via FIBERNET 3.2.0.50 (Div5Net 2/50)
   net 11  Unreachable
   net 43  Unreachable
   net 28  2 hops via DCEC 10.3.0.20 (Arpanet 3/20)
   net 2   1 hop  via R2D2 10.3.0.51 (Arpanet 3/51)
   net 29  2 hops via DCEC 10.3.0.20 (Arpanet 3/20)
   net 35  Unreachable
   net 9   Unreachable
   net 4   Unreachable
   net 25  Unreachable
   net 21  1 hop  via DCEC 10.3.0.20 (Arpanet 3/20)
   net 18  Unreachable
   net 8   Unreachable
```

Note that when this command was given, the BBN gateway was down for preventive maintenance. Consequently the status output showed it down and all the nets beyond it unreachable.

5.5 Macro-11 vs. BCPL Testing

A preliminary test comparing packet throughput between the Macro-11 and BCPL gateways running in the RCC gateway (ARPANET-RCCnet) showed substantially improved performance. The BBN-VAX was used to measure packet and bit throughput by sending TCP packets out its ARPANET interface to its RCCnet interface via the RCC gateway. The packet test was done with the VAX sending small (one TCP data character) packets. The bit throughput test used large (950 TCP data characters) packets. The test consisted of sending 10,000 packets, and each test was repeated several times.

The packet test showed that the BCPL gateway delivered 50 packets/sec while the MACRO-11 gateway delivered 100 packets/sec. The bit throughput test showed both gateways delivering about 20kbits/sec. We believe the second test was limited by the networks, not the gateways.

6 MOBILE ACCESS TERMINAL NETWORK

As part of our participation in the development of the Mobile Access Terminal (MAT) and the MAT Satellite Network (MATNET) during the last quarter, we supported the system integration within the Advanced Command and Control Architectural Testbed (ACCAT) experiment at the Naval Ocean Systems Center (NOSC) in San Diego, California. Although our contract had expired, limiting our support, we were able to participate in January's satellite testing and pass a major milestone in the project. Some of the accompanying events are described below.

Because last September's satellite tests of the MATNET system were plagued with extensive cabling faults (miswired opto-isolators, open grounds, etc.) between the AN/WSC-3 radios and the Black Processors, interference habitually caused the effective satellite channel bit-error-rate to increase to two orders of magnitude beyond the system design limit of 10**-5. The consequent severe packet lossage resulted in the Satellite IMPs declaring themselves out of reservation synchronization an excessive amount of time; hence, no useful information could be gleaned from these satellite tests.

Subsequently, E-Systems, ECI Division, undertook the responsibility of remedying the wiring problems; however, not until January was satellite time again made available for further tests. To provide a convincing demonstration that the cabling problems were indeed eliminated and that the satellite channel was working satisfactorily, we defined the following series of tests. In each of these test setups, packets are generated and processed in the Black Processor, so as to eliminate vagaries with the COMSEC equipment. Tests are divided into separate sessions, each of which is initialized first for short (1 block) packets and afterwards for long (16 block) packets. The tests are differentiated into the three categories below.

Cabling Check:
Each MAT station (one at a time) is connected to the satellite channel simulator with message generators enabled.

Crosstalk Check:
One MAT station is connected to the satellite channel simulator, while one MAT station is transmitting ove the satellite channel with message generators in both MATs enabled. Afterwards the two MAT stations are interchanged (the one to the satellite channel is connected to the satellite channel simulator and vice versa).

Channel Performance:
Both MAT stations are simultaneously receiving satellite channel data, while message generators are enabled in only one MAT station for transmitting over the satellite channel. (The Black Processor does no channel scheduling, which precludes both units transmitting simultaneously.) Afterwards the two MAT stations are interchanged.

Satisfactory performance is indicated by no packet lossage whatsoever when a MAT station is connected to the satellite channel simulator, and a packet lossage corresponding to a maximum effective BER of 10**-5 when a MAT station is connected to the satellite channel. The latter is shown in the set of entries marked Calculated Packet Lossage in the table below. Also shown in the table are the experimental results taken during the January satellite tests, confirming successful operation.

Site	Packet Length (blocks)	Calculated Packet Lossage	Experimental Packet Lossage
Shore	1	0.4%	0.3%
Shore	16	4%	1.1%
Ship	1	0.4%	0.4%
Ship	16	4%	0.8%

Calculated vs. Experimental Packet Lossage

As part of January's satellite testing at NOSC, a major milestone was passed in the MATNET and Command Center Network (CCN) projects; namely, the CCN was configured as a host of the shipboard MAT with connectivity to the ACCAT DEC TOPS-20 via MATNET. A schematic of the configuration is shown below, where G, 1822, LUBPI, NIU and PLI represent a DEC LSI-11/02 gateway, an 1822 Host-to-IMP standard interface, a DEC LSI-11 Ungermann Bass Parallel Interface, a Network Interface Unit, and a Private Line Interface, respectively.

```
+--------+                            +--------+
|        |   1822                     |        |
|TOPS-20 |----------------------------|        |
|        |                            |        |
+--------+                            |        |
                                      |        |
                                      |        |
                                      |  PLI   |
                                      |        |
                                      |        |
+--------+                            |        |
|        |   1822   +---+  1822       |        |
|  MAT   |----------| G |-------------|        |
| SHORE  |          +---+             |        |
+--------+                            +--------+
    |
   / \



   \ /
    |
+--------+                            +--------+
|        |   1822   +---+  LUBPI      |  NIU   |
|  MAT   |----------| G |-------------|  CCN   |
|  SHIP  |          +---+             |  SHIP  |
+--------+                            +--------+
                                          |
                                          |
                                          |
                                       TI-743
                                      TERMINAL
```

CCN/MAT Test Configuration

In the demonstration, a TI-743 terminal attached to an NIU of the shipboard CCN was used to open a TCP connection to the TOPS-20 for accessing files in its directory. MATNET status monitoring data from the TOPS-20 were then printed out on the TI-743 terminal. This demonstration exercises all the fundamental network functions invoked by MATNET and CCN and as such forms a comprehensive test of network operation.

During the last quarter we also fixed a problem that surfaced with the MATNET Operations Center, designated the Network Operations Center or NOC. The basis of NOC is a collection of specially-developed TOPS-20 computer programs, including RECORDER, which process monitoring information received from the Satellite IMPs. Symptoms of the malfunction were that RECORDER, after running for a couple of hours, entered into a state in which it is continually being swapped in and out of memory by the TOPS-20 EXEC operating system. While in this state, TOPS-20 cycles were consumed at a prodigious rate. During a regular RSM (Remote Site Maintenance) at NOSC, we were able to examine RECORDER files for problem diagnosis. To correct the problem, we eliminated from RECORDER the commands which created a 24-hour summary on the disk and mailed the summary to a selected mail list.

Because of the decommission of the ARPANET host computer BBN-TENEX-E, we were required to move the MATNET project directories from BBN-TENEX-E to BBN-TENEX-C. The long-term goal, however, is to transfer the project directories to a UNIX computer. Toward that goal we have begun the conversion of Satellite IMP source software to a form compatible with a C/30 cross assembler currently residing in BBN-UNIX machines. Backroom debugging of this assembled software has occupied a significant amount of our time.

7 TCP FOR THE HP3000

The last quarter of the HP3000 Internet project was spent on two efforts. First, we continued our testing of existing software and cleared up some problems; the tests included the development on an Internet Name Server to test the internet datagram interface developed last quarter. Second, we designed a network routing table which will handle the new netw
Broadcast Message Fromork
addressing scheme.

7.1 Name Server

The Name Server was created as a test of the Internet protocol software. Specifically, we were interested in testing the concept of name servers as well as testing the response time of our raw internet datagram interface. Both the speed and the utility of the name server will become increasingly important as new networks are added to the Catenet. As the number of addressable hosts increases it will become more and more difficult for small hosts to maintain the databases necessary to address all of the hosts. Our effort was intended as a first order test of the Name Server concept.

The HP3000 name server program is implemented as a user level program in accordance with IEN 116, which describes the

Internet Name Server and its protocols. The program combines the User Datagram Protocol (UDP), described in RFC 768, and the Internet Protocol to set up a UDP listen for all datagrams which contain requests for the name server. The User Datagram Protocol is identified by the value 17 in the protocol ID field of the Internet header. The Name Server uses port 42 in the UDP protocol. The protocol and port number values were chosen in accordance with IEN 127, Assigned Numbers.

The name server [illegible]am takes the requests, checks them for authenticity and consistency, searches its database for the correct host address(es), and sends a reply to the host requesting the name service. The replies are either error messages if the requested data is not in the database, or the host names and addresses if the requested data was found in the database.

So far, name server tests from the HP3000 to itself show that there are no significant delays incurred in name server requests. Delays of about 1 second were common. While this initial result is encouraging, more tests are needed using hosts other than the HP3000.

7.2 New Network Routing Table Design

Upcoming changes in the Internet Protocol (IP) have caused us to redesign the network table in the TCP/IP process. The format for addresses in the Internet Protocol has been changed to allow for a vastly larger number of networks than the 256 networks allowed by the old format. The new IP format allows in excess of two million networks.

The TCP/IP process keeps a table in memory which, for each non-local network, indicates the appropriate gateway to use when sending messages. Currently, this table has one entry for each of the possible 256 networks, and accesses are very fast. However, it is not feasible to have one entry in memory per network for each of two million networks, so the table has had to be redesigned.

There is an operational constraint that we have been able to use in guiding the new table design: the table need only be large enough to hold data for those networks with which our host is in active communication. All outgoing datagrams originate in user programs, and the number of network connections available on the system is limited to the compile time constant NCON (currently 128). If the program establishes a TCP network connection, its destination network is constant for the duration of the TCP connection. In an IP connection, it is possible for each datagram to be for a different network, but in any given interval it has only a small number of networks it can possibly address.

With this in mind, we have designed the new table to have slightly more than NCON entries. Entries in the new table must now contain the network number, in addition to the gateway address to use and related information. On the average, there will be one network entry per connection. In practice, it is likely that many connections will be to the same network, leaving room for any connections that might wish to access several networks concurrently.

The table will be arranged with network numbers monotonically increasing. One option, which we ultimately decided against was to maintain the table as a compacted list of networks in use. New network addresses could be sorted into the correct place, moving down higher numbered networks, and removing unused network entries as required. However, this simple scheme would mean the table would have to be searched for the entry of the desired network for each outgoing datagram. Neither a linear nor a binary search of the entire table would be very efficient compared to the direct lookup currently in use.

To speed up the search, we chose to divide the table into two separate sections. In the lower section, there will be an entry for every network number from zero to some upper bound. In the second section, network entries will be added and deleted as needed and a search will have to be made to find the correct entry.

By placing the currently defined networks (numbers 1 through around 40) in the first section, we regain the speed of a single direct lookup that our current code enjoys. In foreseeable practice, these networks will also be the most commonly used networks. By confining the networks to be searched to a much smaller table, we also reduce the search time needed to look up an entry in the second section. Our current estimate is that a lookup in the second section will take only three probes, with five maximum.

To initialize this table, we will continue to use a file that gives network/gateway pairs for networks in the first section, and we will initialize the second section to indicate all entries that are unused. An entry is added to the second section when a datagram appears that must be sent to a network not already in the table. The first datagram to that network will be sent to a selected smart gateway, and that gateway's address will be the initial value for the network table entry. For any network in the table, if a gateway sends a REDIRECT notification, the network entry will be updated. Thus entries in the first section will be updated if they are wrong, and network entries in the second section will be quickly set to the correct gateway.

8 TCP FOR VAX-UNIX

This quarter's work on the VAX TCP project centered on three areas: distribution to beta-test sites, maintenance and enhancement of the TCP/IP kernel software, and maintenance and enhancement of the higher level protocol and support software. The funding gap we had experienced at the end of the last quarter ended, and work on the project recommenced in early December.

8.1 Software Distribution

8.1.1 Distribution to Test Sites

A version of the TCP/IP implementation has been shipped to eight sites, including Berkeley, CMU, Purdue, Stanford, MIT, USC-ISI, University of Wisconsin, and CalTech. The software is installed and running at most of these sites. Support of the test sites included consultation on installation problems, and bug isolation and repair. To this end, three sets of updates and bug fixes have been circulated among the test sites to date. In addition, installation procedures have been refined to a point where the process needs little assistance from us. We are concentrating on making this procedure as trouble-free as possible, in anticipation of the general release. The main problems we have encountered at the test sites have resulted from improper installation of the software and interactions with local (test site) modifications to UNIX.

8.1.2 General Distribution

We are planning to make the first general release of the software in the coming quarter. Before this can be done, several problems must be answered. Currently, modifications have been made to an earlier version of our implementation at Berkeley which have resulted in a version that is significantly different from our own. The Berkeley version includes a revamped user interface that is part of their general interprocess communication (IPC) scheme, systems performance enhancements oriented toward high speed local networking, significant reorganization of the TCP itself, and modifications that result in serious departures from the protocol specifications. Some of these modifications will be easily integrated with our implementation (new user interface, certain performance enhancements). However, we believe that some of them (reorganization of the TCP, and performance enhancements that require departure from protocol specifications) are inappropriate for general distribution. In addition, those features that we do intend to adopt (such as the IPC user interface), are not yet adequately defined for use in networking, and will require further discussion.

It is clear that some reconciliation of these differences is required. One solution would be to prepare a version suitable for general release in the near term, and negotiate the form of a version of the networking code that will be part of the next major release from Berkeley (Fall 1982). This will require a mechanism for general distribution of the near-term release (by BBN). In addition, it will entail some inconvenience for users of the software when the next Berkeley software distribution is made. It is hoped that these issues will be resolved at the upcoming February meeting of the Berkeley Steering Committee.

8.2 TCP/IP Enhancements

Several significant enhancements were made to the TCP/IP kernel implementation, in addition to the ongoing testing and maintenance work. These include completion of ICMP and GGP message handling code, addition of a UDP (User Datagram Protocol) implementation, completion of a revamped Direct Message Interface that replaces the raw IP and local network protocol interface, a driver for the Proteon Associates PRONET (V2LNI) Local Network Interface, and performance enhancements for local networking.

8.2.1 Direct Message Interface

The mechanism for accessing the IP and local network layers has been redesigned to allow users to register for simultaneous multiple access to a group of dispatch numbers (e.g., IP protocol or ARPANET 1822 link numbers). For example, two or more users may wish to receive all IP messages with the TCP protocol number. The system would then copy all TCP messages to the requestors, as well as to the internal TCP module. In addition, users can specify an array of dispatch numbers, and add to or delete from the array.

These facilities have been implemented using a scheme that is designed to minimize the overhead on the "built in" protocols (TCP, UDP, IP, ICMP, and GGP) when they are not in use. This is done by maintaining a hashed table of pointers to a linked list of "protocol blocks". For each protocol registered for in a connection, a protocol block is allocated in an array associated with the connection block (UCB), and linked into the list for that dispatch number. To reduce overhead on the built in protocols, a special list head entry is reserved for each, thus obviating the hashing step for them. When a message arrives for one of the built-in protocols, the header entry is checked, and if it is non-zero (there are other requestors), the message is copied. The original message is then sent to the built-in protocol module, and then to the direct message dispatcher, which places copies of the message on each of the requestors' queues. When there are no requestors for a built-in protocol, the only overhead incurred is a test of the protocol block header for that protocol. Otherwise, the only overhead is a single copy operation. For other protocols which are not built-in, the process is similar, except that the protocol block list header is found by hashing on the dispatch number.

Sending messages works much like the old scheme. The user may opt to construct the protocol headers or have the system do it, based on information in the connection block. With multiple dispatch numbers, a "primary" number (the first in the array) is used in system-constructed outgoing message headers.

Error handling has also been refined. On receiving, messages that exceed the user's queue limit are discarded, but an indication that messages were dropped is given to the user. On sending, messages are accepted only if the send queue limit is not exceeded and the message can be transmitted to the network. If the network interface is blocked, an "interface blocked" error results. Special local network processing, such as ARPANET RFNM counting, is now done for direct messages, as well as for TCP.

One test of this facility was to move the internal IP fragment reassembly code to a user level program for debugging purposes, and operate it in parallel with the internal IP, using the direct message interface. This approach is useful for testing the fragment reassembly code, as debugging printouts can be inserted easily in the user level code, and debugging tools can be used while the system is running. Another advantage is that "test data" for the user level program is actual network traffic. With the direct message interface, other sections of the network code, such as the entire TCP module itself, can be tested as user programs.

8.2.2 Local Networking Support

A driver for the Proteon Associates PRONET 10Mbaud ring was designed and debugged. This hardware is in use at several of the test sites (MIT, Purdue, University of Wisconsin). We tested and debugged the driver using interface hardware on loan from the manufacturer.

As part of this effort, we have been looking into performance issues relating to high speed local networks. Currently, the highest looping throughputs we have seen with local network interfaces have been on the order of 275Kbits/sec, with both 3Mbaud prototype Ethernets and the PRONET. We are working on adopting several performance enhancements suggested by Berkeley. New assembly language coded byte swapping and checksum routines have already been adopted. We are examining tuning changes such as scaling timer values to make them more suited to local networking speeds.

In addition, we are planning to reorganize the software to make the entire networking code run at interrupt level. This should simplify the code, and reduce scheduling overhead significantly.

8.3 Higher Level Protocol Support

As part of the support for the higher level protocols (Telnet, FTP, and MTP), which are implemented in user level software, we are working on a new version of the network host name/address library software. The new network library recognizes internet format addresses (four decimal digits representing the bytes of the internet address, separated by periods). In addition, it allows strings representing host attributes to be stored and retrieved. This is necessary for the NCP/TCP transition, and for mail applications. The new network library also gives better support for multi-homed hosts. It is implemented as a hashed database of ASCII strings representing internet addresses, host names, and attribute keywords. Any binary conversions are done on the fly. Entries are indexed by host name, aliases, and address, for fast retrieval. Facilities for creating the database from several different sources are available.

The new network library software will be used in an experimental implementation of an internet name server. In addition, we are implementing software to maintain the name database for individual hosts. The envisioned mode of operation is that the hosts will maintain a cache of frequently used network addresses and use the name server to build the cache.

DISTRIBUTION

<u>ARPA</u>
Director (3 copies)
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209
Attn: Program Manager
R. Kahn
V. Cerf
R. Ohlander
D. Adams

<u>DEFENSE DOCUMENTATION CENTER</u> (12 copies)
Cameron Station
Alexandria, VA 22314

<u>DEFENSE COMMUNICATIONS ENGINEERING CENTER</u>
1860 Wiehle Road
Reston, VA 22090
Attn: Lt. Col. F. Zimmerman

<u>DEPARTMENT OF DEFENSE</u>
9800 Savage Road
Ft. Meade, MD 20755
R. McFarland R17 (2 copies)
M. Tinto S46 (2 copies)

<u>DEFENSE COMMUNICATIONS AGENCY</u>
8th and South Courthouse Road
Arlington, VA 22204
Attn: Code 252

<u>NAVAL ELECTRONIC SYSTEMS COMMAND</u>
Department of the Navy
Washington, DC 20360
B. Hughes, Code 6111
F. Deckelman, Code 6131
J. Machado, Code 6134

<u>BOLT BERANEK AND NEWMAN INC.</u>
1701 North Fort Myer Drive
Arlington, VA 22209
E. Wolf

<u>MIT Laboratory for Computer Science</u>
545 Technology Square
Cambridge, MA 02138
D. Clark

DISTRIBUTION cont'd

<u>BOLT BERANEK AND NEWMAN INC.</u>
10 Moulton Street
Cambridge, MA 02238

M. Brescia
R. Bressler
R. Brooks
P. Carvey
P. Cudhea
W. Edmond
L. Evenchik
G. Falk
J. Goodhue
S. Groff
R. Gurwitz
J. Haverty
F. Heart
J. Herman
R. Hinden
D. Hunt
E. Hunter
S. Kent
A. Lake
W. Mann
A. McKenzie
D. McNeill
W. Milliken
A. Nemeth
R. Rettberg
J. Robinson
E. Rosen
G. Ruth
P. Santos
J. Sax
A. Sheltzer
E. Starr
S. Storch
R. Thomas
B. Woznick
Library